機器ドライバ
ガイドブック
– ActiveX 編 –

Hiroshi Yamaguchi

Dec 2000　2nd edition

KIKUSUI
菊水電子工業株式会社

はじめに

1. ソフトウェア使用許諾契約

下記の使用許諾契約の各条項は当社 WEB のダウンロードページに記載されているものです。 既にダウンロードして、このガイドブックをお読みになっている貴方様は本使用許諾契約に同意したことになります。 本使用許諾契約に同意いただけない場合は、すぐに削除してください。

1. 権利の許諾

菊水電子工業株式会社（以下「当社」といいます）はお客様に対して、本使用許諾契約に同意いただいてダウンロード可能となるソフトウェア及びその関連資料（以下「本ソフトウェア」といいます）に関し、以下の権利を許諾します。

（a）お客様は、本ソフトウェアに対応する当社製品を利用する目的で本ソフトウェアを使用することができます。

（b）お客様は、本ソフトウェアを複製し、1台以上のコンピュータ上で使用することができます。

2. 追加許諾条項

本ソフトウェアを定められた目的に従って使用した結果、作成された各種のファイルは、お客様の著作物となります。

3. 著作権

本ソフトウェア及びその複製物の著作権は当社又は当社が認めた者が有するものであり、日本国著作権法及び国際条約によって保護されています。 本使用許諾契約に基づき、お客様が本ソフトウェアを複製する場合は、ダウンロードされた本ソフトウェアに付されていたものと同一の著作権表示がなされることを要します。

4. 禁止事項

（a）本ソフトウェアがソースコードで提供される場合、お客様は、本ソフトウェアを改変したものを第三者に配布することはできません。

（b）本ソフトウェアがバイナリ形式で提供される場合、お客様は、本ソフトウェアをリバースエンジニアすることはできません。

5. 無保証

当社は、本ソフトウェアがお客様の特定の目的のために適当であること、もしくは有用であること、又は本ソフトウェアに瑕疵がないこと、その他本ソフトウェアに関していかなる保証もいたしません。

6. 免責

当社は、いかなる場合においても、本ソフトウェアの使用又は使用不能から生ずるいかなる損害（事業利益の損害、事業の中断、事業情報の損失、又はその他金銭的損害）に関して、一切責任を負いません。

7. 契約の解除

お客様が本使用許諾契約に違反した場合、当社は本使用許諾契約を解除することができます。その場合、お客様は本ソフトウェアを一切使用しないものとします。

2. サポート

各ソフトウェアについてのお問い合わせは、当社サービス（service@kikusui.co.jp）までお願いたします。

3. 保証範囲

ソフトウェアのダウンロード・インストールはお客様の責任において行っていただきます。

ソフトウェアは、予告せず改良、変更することがあります。

4. 著作権者

本ガイドブックおよび各ソフトウェアの著作権は、菊水電子工業株式会社に帰属します。

5. 本ガイドブックについて

当社ホームページ ダウンロード・サービスの Visual Basic（ActiveX）ドライバの導入部分についてサンプルを挙げながらガイドするものです。 ダウンロードはしてみたが、どうすればよいのかな？ という最初のとっかかりに困った方はこのガイドブックをお読みください。

Visual Basic 6.0, Excel 97, Visual C++ 6.0, Delphi 3.0 を使用した具体例を各章ごとに分けて解説しますので、お使いになるツールの章だけを読めば良いように構成してあります。はしがきから読んでいる律義な貴方には重複した部分が多く感じるかも知れません。

以後、本ガイドブックでは「Visual Basic（ActiveX）ドライバ」のことを単に「機器ドライバ」といいます。

6. 各社商標

Borland, Delphi は Inprise Corporation の商標または登録商標です。

IBM は International Business Machines Corporation の登録商標です。

Microsoft, Windows, Visual Basic, Visual C++, ActiveX は米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。

National Instruments, NI-488.2, NI-VISA は National Instruments Corporation の商標または登録商標です。

その他、記載されている会社名、製品名は各社の商標または登録商標です。

目次

第 1 章 - 概要 5
1.1 機器ドライバとは 5
1.2 ツール 5
1.3 動作環境 5
第 2 章 - 導入とセットアップ 6
2.1 インターフェイスカード 6
2.2 GPIB ボード 6
2.3 インターフェイスケーブルの接続 6
2.4 機器ドライバのインストール 6
2.5 ツール 6
第 3 章 - VISUAL BASIC6.0 7
3.1 コンポーネントの追加 7
3.2 Form へのコントロールの貼付け 7
3.3 最初のコード - Connect 8
3.4 Help の参照 9
3.5 プロパティページ 9
3.6 イベント処理 10
3.7 サンプルコード 10
3.8 SetString メソッド, GetString メソッド 11
第4章 - EXCEL 97 12
4.1 コントロールの追加 12
4.2 UserForm の追加とコントロールの貼り付け 12
4.3 最初のコード - Connect 13
4.4 Help の参照 14
4.5 プロパティページ 14
4.6 イベント処理 15
4.7 サンプルコード 15
4.8 SetString メソッド, GetString メソッド 17
第 5 章 - VISUAL C++ 6.0（ダイアログベースのアプリケーション） 18
5.1 プロジェクトの作成 18
5.2 コンポーネントの追加 19
5.3 コントロールの配置 20
5.4 最初のコード - Connect 21
5.5 イベントの処理 21
5.6 サンプルコード 22
5.7 ビルドして実行 22
5.8 Help 22
第 6 章 - VISUAL C++ 6.0（事前バインディングによる方法） 23
6.1 プロジェクトの作成 23
6.2 タイプ ライブラリのインポート 23
6.3 メニューの追加 24
6.4 データメンバの追加 24
6.5 コンストラクタへの記述 25
6.6 コネクトメニューのコード 25
6.7 ビルドして実行 25
第 7 章 - DELPHI 3.0 26
7.1 コントロールの取り込み 26
7.2 Form へのコントロールの貼付け 27
7.3 最初のコード - Connect 27
7.4 try - except による例外処理 27
7.5 イベント処理 28
7.6 サンプルコード 28

第 1 章 - 概要

1.1 機器ドライバとは

当社ホームページでダウンロードサービスを行っている機器ドライバは、菊水製品を GPIB や RS-232C を利用してアプリケーションプログラムを作成するための便利な開発キットです。

機器ドライバは COM（Component Object Model）技術をベースに作られています。 Visual Basic や Microsoft Office あるいは Delphi 等の ActiveX に対応した言語ツールでご使用になれます。

たとえば、Visual Basic ユーザが機器ドライバを使わずに GPIB 制御のアプリケーションソフトを作成する場合、National Instruments 製の gpib-32.dll をリンクし、NI の関数表と菊水のコマンド表を調べながらコードを記述しなければなりません。

機器ドライバを使えば、Visual Basic 付属のボタンやリストボックスのように、フォームに直接貼り付けられ、読みやすいコードで記述することができます。

1.2 ツール

本ガイドブックでは Visual Basic 6.0, Excel 97, Visual C++ 6.0, Delphi 3.0 を使用した具体例を挙げて解説します。

他のツールをお使いの方へ：
32bit 版で ActiveX をサポートしているものであれば、他のツール（Visual Basic 4.0 / 5.0, C++ Builder 等）でもご使用になれます。

1.3 動作環境

VISA（Virtual Instrument Software Architecture）ライブラリは GPIB と RS-232C のどちらをお使いになる場合でも必要です。 この機器ドライバは VISA 2.0 に対応しています。 NI-VISA では動作確認を行っていますが、HP-VISA 等では確認をしていません。 VISA ライブラリは GPIB ボードや HP-IB ボードに同梱されていることもあります。 また、LabVIEW などの CD-ROM に同梱されています。 NI-VISA 2.0 は http://www.ni.com/visa/ からダウンロードすることができます。

GPIB ユーザ:

Windows 9x/NT4/2000 が動作するパソコン

National Instruments NI-488.2M 互換ボード または Agilent Technologies HP-IB ボード

GPIB ケーブル

RS-232C ユーザ:

Windows 9x/NT4/2000 が動作するパソコン

RS-232C クロスケーブル

下記バージョンの機器ドライバについては VISA ライブラリに対応していませんので、最新ドライバをダウンロードしてください。

PCR-L/W シリーズ	Ver 2.70 以前
PLZ-3W/3WH シリーズ	Ver 1.00 以前
PIA3200 シリーズ	Ver 1.10 以前
TOS9000 シリーズ	Ver 1.01 以前
KJM6755A シリーズ	Ver 1.00 以前

PCR-L/W シリーズ Ver2.70 の機器ドライバはツールへの組込み方法が若干異なります。 本ガイドブックより詳しいドキュメントが添付してありますので、そちらのドキュメントをお読みください。
PCR-L/W シリーズ Ver2.70 の機器ドライバでは GPIB ボードは National Instruments NI-488.2M 互換ボードにのみ対応します。

第 2 章 - 導入とセットアップ

2.1 インターフェイスカード

GPIB または RS-232C を標準装備していない機器をコントロールする場合はオプションのインターフェイスカードを機器に装着する必要があります。 適応するオプションカードについては機器の取扱説明書を参照してください。 カードを装着したら機器のアドレスを設定してください。 RS-232C の場合は通信パラメータや接続ケーブルが機種ごとに異なりますので、ご注意ください。

2.2 GPIB ボード

GPIB を使用する場合は、National Instruments NI-488.2M 互換ボード または Agilent Technologies HP-IB ボードが 1 枚、パソコンに必要です。

また、パソコンには VISA ライブラリをインストールしておく必要があります。

2.3 インターフェイスケーブルの接続

機器とパソコンの両方のインターフェイスカードが装着できたら、ケーブル（IEEE488 ケーブルまたは RS-232C クロスケーブル）で接続してください。

2.4 機器ドライバのインストール

ダウンロードした機器ドライバ xxx.exe は自己解凍型のインストーラとなっています。 名前を指定して実行するかエクスプローラでダブルクリックします。 Figure 2-1 のようなダイアログが開きますので、[Setup] ボタンをクリックし、あとは画面の指示に従ってください。

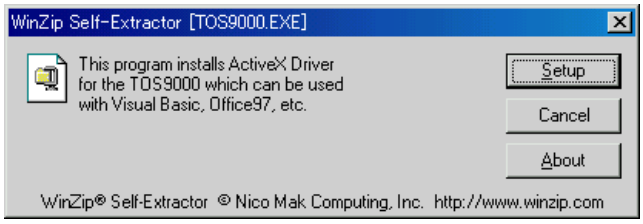

Figure 2-1 自己解凍型インストーラ

Setup Complete ! と表示されたら [Close] ボタンをクリックして画面を閉じればインストールは完了です。 インストーラは C:¥Program Files¥Common Files¥Kikusui Shared フォルダに必要なファイルをコピーし、レジストリに登録します。

2.5 ツール

ご使用になる Visual Basic 6.0, Excel 97 等はあらかじめインストールしてあることとします。

本ガイドブックはご使用のツールの章だけ読めば良いように構成されています。

第 3 章	Visual Basic 6.0
第 4 章	Excel 97
第 5 章	Visual C++ 6.0（ダイアログベースのアプリケーション）
第 6 章	Visual C++ 6.0（事前バインディング）
第 7 章	Delphi 3.0

必要な章だけお読みください。

第 3 章 - Visual Basic6.0

3.1 コンポーネントの追加

Visual Basic 6.0 を起動し、[新しいプロジェクト]ウィンドウから 標準 EXE を選択します。Visual Basic の[プロジェクト]メニューから[コンポーネント...]を選択すると Figure 3-1 のコンポーネント ダイアログボックスが表示されます。機器ドライバがインストールされていれば、そのリストに **□ Kikusui xxxx Driver Vx.x** が表示されているはずです。リストから目的のデバイスの機器ドライバが見つかったらチェックしてください。 OK ボタンをクリックしてダイアログを閉じると[ツールボックス]上に機器ドライバのコンポーネントが追加されます。（Figure 3-2）

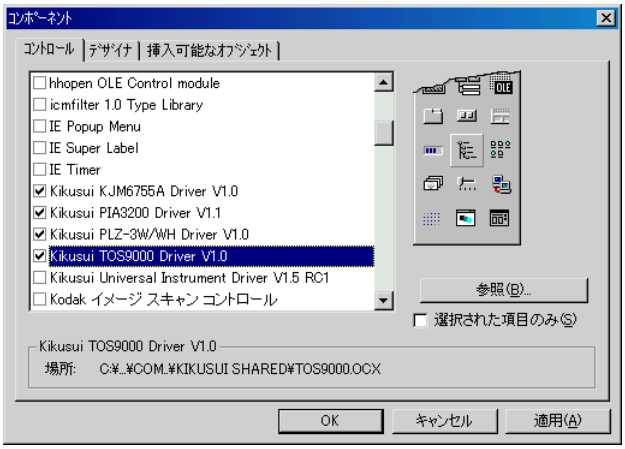

Figure 3-1　Visual Basic 6.0 のコンポーネントダイアログ

3.2 Form へのコントロールの貼付け

[ツールボックス]から Command Button と Tos9000 を Form へ貼り付けます。プロパティウィンドウで Tos9000 のデフォルトのオブジェクト名 Tos90001 を tos と代えておきます。もちろん、デフォルトのままでも OK です。

注記: ここでは TOS9000 の機器ドライバを例にとりますが、他の機器ドライバをお使いになる場合でも、容易に想像可能と考えます。

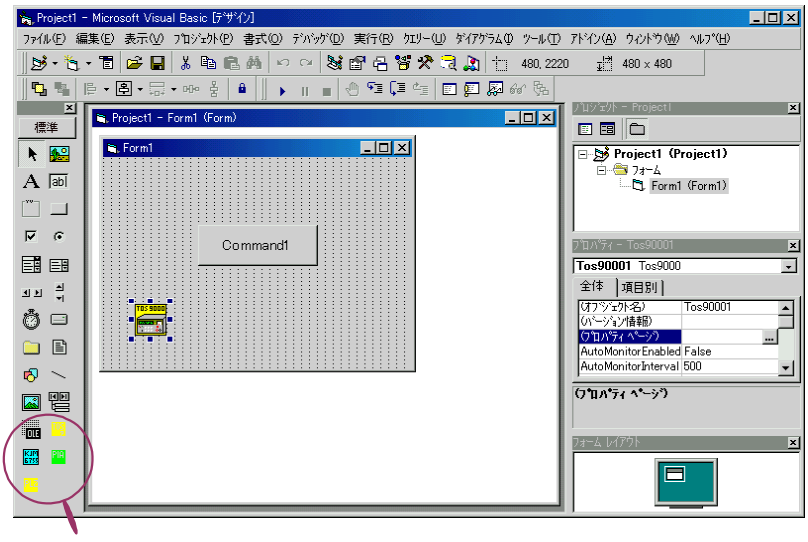

Figure 3-2　Visual Basic 6.0 のツールボックスとフォーム

3.3 最初のコード - Connect

Command1 ボタンをダブルクリックするとコードが表示され、Command1 ボタンをクリックしたときの処理が記述できます。 Tos9000 のオブジェクト名 tos. と入力すると Visual Basic の自動メンバ表示機能により、Tos9000 オブジェクトのメンバが表示されます。（Figure 3-3） まず、Connect メソッドにカーソルを合わせて Ctrl + Enter と入力します。すると tos. に続けて Connect と入力できます。

> 注記: Visual Basic の[ツール]メニュー →[オプション]→[編集]タブで、自動メンバ表示機能の ON/OFF が設定できます。

さらに Space を入力すると Connect(DevName As String) とガイドが表示されます。ガイド通りに String 型でデバイス名 "GPIB::1" と入力します。 DevName "GPIB::1" は GPIB アドレスが 1 の場合です。 RS-232C では 'ASRL1' と入力します。

```
Private Sub Command1_Click()
    tos.Connect "GPIB::1"
End Sub
```

この Connect メソッドは文字通り「接続する」という意味でコードの最初に記述してください。この１行だけのプログラムを実行して Command1 ボタンをクリックすると TOS9000 は REMOTE 状態になるはずです。 反対に「切り放す」は Disconnect です。

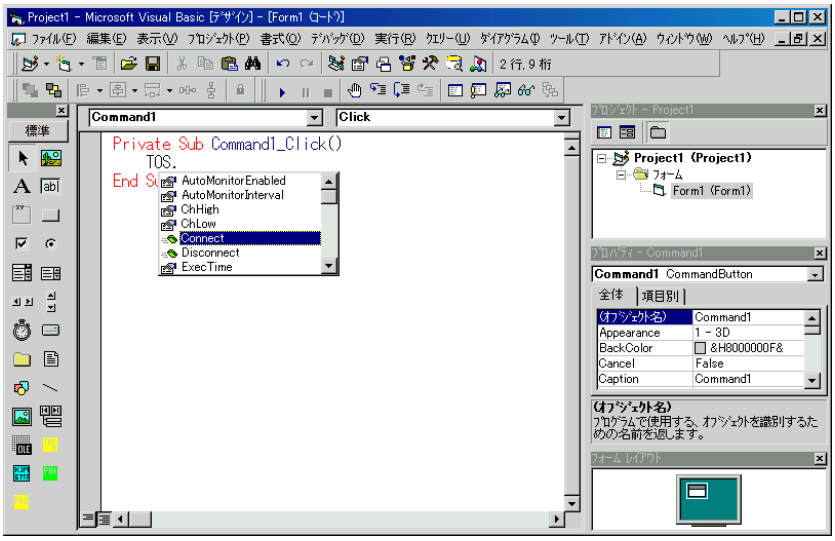

Figure 3-3 自動メンバ表示機能

従って、下記のコードのように Form が Load されるときに Connect して、Unload されるときに Disconnect するのも一つの方法です。 また、Command ボタンを追加し、Caption をかえて Connect ボタン, Disconnect ボタンを付けるのもよいと思います。

```
Private Sub Form_Load()
    tos.Connect "GPIB::1"
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    tos.Disconnect
End Sub
```

Connect までできれば、要領が把握できたと思います。 Help を参照しながらアプリケーションを完成させてください。

3.4 Help の参照

機器ドライバの Help ファイルの参照の仕方は、Visual Basic 6.0 の表示メニューから オブジェクト ブラウザ を表示し、プロジェクト/ライブラリ の ComboBox で "TOS9000LibCtl" を選択します。 Tos9000 のメンバのどれかにカーソルを合わせて ヘルプ ボタンをクリックすると TOS9000 Driver の Help が表示されます。

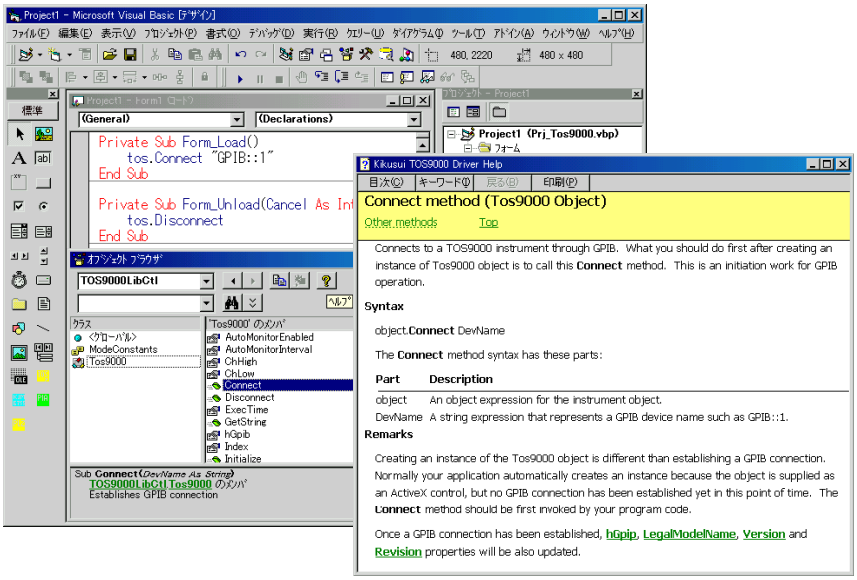

Figure 3-4　オブジェクト ブラウザによる Help ファイルの参照

3.5 プロパティページ

また、TOS9000 の機器ドライバには設計時プロパティが備わっていますので、これを活用すれば Start ボタンと Stop ボタンがあれば制御できます。 Auto Monitor タブはイベント処理のためのプロパティの設定です。

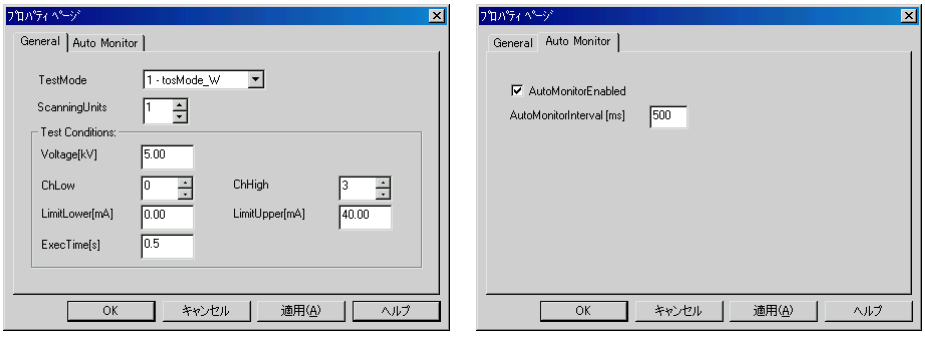

Figure 3-5　TOS9000 ドライバのプロパティページ

3.6 イベント処理

Form 上の TOS9000 アイコンをダブルクリックするか、コード表示の最上部のオブジェクト ComboBox を tos にして、プロシージャ ComboBox を TestingW にします。(Figure 3-7)
すると、W テスト中のイベント処理を記述する Sub プロシージャが生成されます。

```
Private Sub tos_TestingW(ByVal Status As Integer, ByVal Voltage As Double, ByVal Current As Double)


End Sub
```

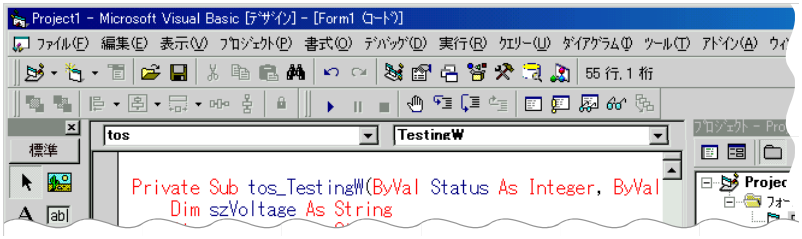
Project1 - Microsoft Visual Basic [デザイン] - [Form1 (コード)]
tos
TestingW
Private Sub tos_TestingW(ByVal Status As Integer, ByVal
Dim szVoltage As String

Figure 3-6　Visual Basic のオブジェクトとプロシージャの ComboBox

このイベントを有効にするには AutoMonitorEnabled プロパティを True にする必要があります。 AutoMonitorInterval[ms]を適当な値に設定してください。 500ms 程度が良いでしょう。
3.7 項のサンプルコードでは耐電圧試験の試験中の電圧と電流を 500ms おきに Label に表示します。

3.7 サンプルコード

Setting ボタンで TestMode や TestVoltage を設定し、Start ボタンで開始、Stop ボタンで停止するサンプルプログラムです。 Form のロード時に Connect し、アンロード時に Disconnect します。 Connect でエラーが発生するとメッセージボックスを出します。

Monitor ボタンは MonVoltage プロパティでテスト電圧を取得し Label に表示します。 イベント処理のサンプルと重複しますが、Monitor ボタンはテスト中以外でも有効です。

```
Private Sub Form_Load()                         ' Form生成時の処理
    On Error GoTo GPIB_EXP                      ' 例外が発生した時GPIB_EXPへ

    tos.Connect "GPIB::1"                       ' ｺﾈｸﾄ
    Exit Sub

GPIB_EXP:
    MsgBox Err.Description                      ' 例外発生時のﾒｯｾｰｼﾞﾎﾞｯｸｽ
    tos.Disconnect
    End                                         ' ｱﾌﾟﾘｹｰｼｮﾝの終了
End Sub

Private Sub Form_Unload(Cancel As Integer)' Form消滅時の処理
    tos.Disconnect                              ' ﾃﾞｨｽｺﾈｸﾄ
End Sub

Private Sub Monitor_Click()                     ' Monitorﾎﾞﾀﾝがｸﾘｯｸされた
    Label1.Caption = Format(tos.MonVoltage, "0.00kV")
                                                ' Label1に今の電圧値を表示
End Sub

Private Sub Setting_Click()                     ' Settingﾎﾞﾀﾝがｸﾘｯｸされた
    tos.TestMode = tosMode_W                    ' 耐圧試験ﾓｰﾄﾞ
    tos.ScanningUnits = 0                       ' ｽｷｬﾅなし
    tos.Voltage = 5                             ' 試験電圧 5kV
    tos.LimitLower = 0                          ' 下限設定 0mA
    tos.LimitUpper = 40                         ' 上限設定 40mA
    tos.ExecTime = 10                           ' 試験時間 10sec
End Sub
```

```
Private Sub Start_Click()                    ' Startﾎﾞﾀﾝがｸﾘｯｸされた
    tos.Start                                ' ｽﾀｰﾄ
End Sub

Private Sub Stop_Click()                     ' Stopﾎﾞﾀﾝがｸﾘｯｸされた
    tos.Stop                                 ' Stop
End Sub

Private Sub tos_TestingW(ByVal Status As Integer, ByVal Voltage As
Double, ByVal Current As Double)             ' ｲﾍﾞﾝﾄ処理
    Dim szVoltage As String
    Dim szCurrent As String

    If Status = 2 Then
        szVoltage = Format(Voltage, "0.00kV") ' 電圧値を文字列に変換
        szCurrent = Format(Current, "0.00mA") ' 電流値を文字列に変換
        Label2.Caption = szVoltage + Chr(&HD) + Chr(&HA) + szCurrent
                          // Label2に 電圧値 + 改行 + 電流値 で表示
    End If
End Sub
```

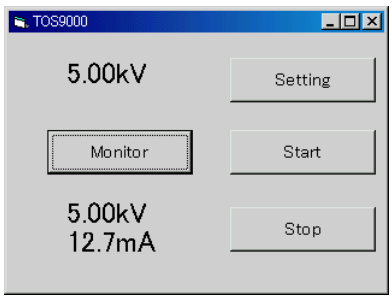

Figure 3-7　サンプルプログラムの実行

3.8 SetString メソッド, GetString メソッド

SetString メソッドはコマンドをそのまま扱えます。 ibwrt と ibrd あるいは PRINT@1; と INPUT@1; のようなものです。

```
Private Sub Monitor_Click()
    tos.SetString "VDATA?"
    Label1.Caption = tos.GetString
End Sub
```

第4章 - Excel 97

4.1 UserForm とコントロールの追加

Excel97 起動し［表示］メニューの［ツールバー］から［Visual Basic］を選択して Visual Basic ツールメニューをアクティブにしてください。ドラッグしてメニューの中に入れてしまうと良いでしょう。Visual Basic ツールメニュー上で、Visual Basic Editor というツールチップが表示されるボタンをクリックしてください。Microsoft Visual Basic – Book1 というウィンドウ（以後、Excel VBA または、単に VBA といいます）が開きます。VBA の［挿入］メニューから［ユーザーフォーム］を選択して新しいフォームを追加してください。VBA の［ツール］メニューから［その他のコントロール...］を選択すると Figure 4-1 のコントロールの追加ダイアログボックス表示されます。機器ドライバがインストールされていれば、そのリストに **☐ Kikusui xxxx Driver** が表示されているはずです。リストから目的のデバイスの機器ドライバが見つかったらチェックしてください。 OK ボタンをクリックしてダイアログを閉じると［ツールボックス］上に機器ドライバのコンポーネントが追加されます。（Figure 4-2）

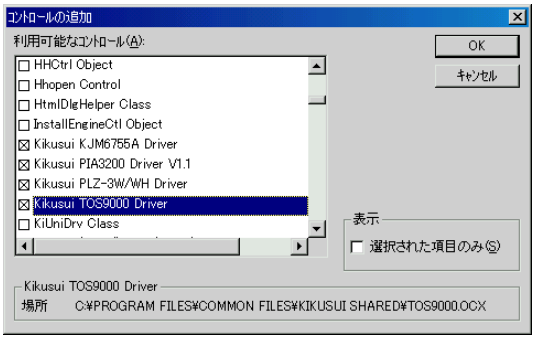

Figure 4-1　Excel VBA のコントロールの追加ダイアログ

4.2 コントロールの貼り付け

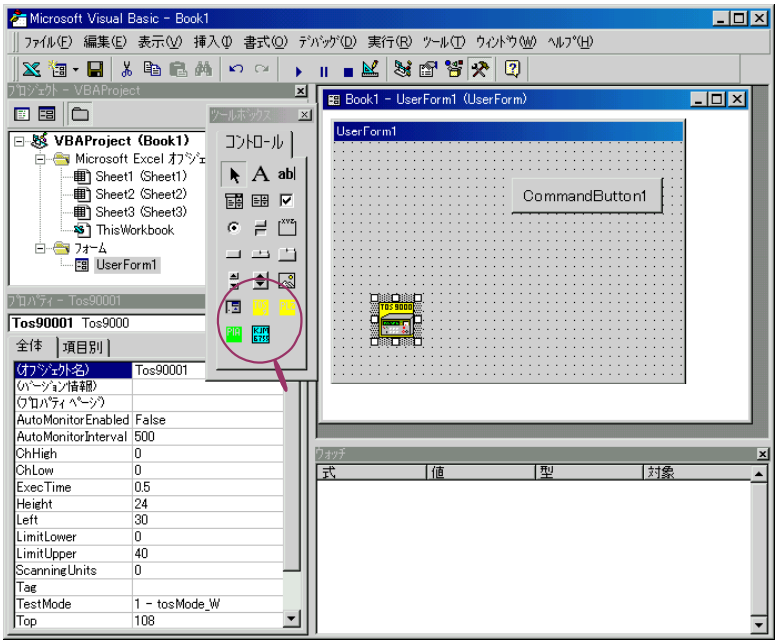

Figure 4-2　Excel VBA のツールボックスとユーザフォーム

Figure 4-2 のように［ツールボックス］から コマンドボタン と Tos9000 を UserForm に貼り付けます。プロパティウィンドウで Tos9000 のデフォルトのオブジェクト名 Tos90001 を tos と代えておきます。もちろん、デフォルトのままでも OK です。

注記: ここでは TOS9000 の機器ドライバを例にとりますが、他の機器ドライバをお使いになる場合でも、容易に想像可能と考えます。

4.3 最初のコード - Connect

CommandButton1 ボタンをダブルクリックするとコードが表示され、CommandButton1 をクリックしたときの処理が記述できます。 Tos9000 のオブジェクト名 tos. と入力すると Visual Basic の自動メンバ表示機能により、Tos9000 オブジェクトのメンバが表示されます。（Figure 4-3） まず、Connect メソッドにカーソルを合わせて Ctrl + Enter と入力します。すると tos. に続けて Connect と入力できます。

注記: VBA の[ツール]メニュー →[オプション]→[編集]タブで、自動メンバ表示機能の ON/OFF が設定できます。

さらに Space を入力すると Connect(DevName As String) とガイドが表示されます。ガイド通りに String 型でデバイス名 "GPIB::1" と入力します。 DevName "GPIB::1" は GPIB アドレスが 1 の場合です。 RS-232C では 'ASRL1' と入力します。

```
Private Sub CommandButton1_Click()
    tos.Connect "GPIB::1"
End Sub
```

この Connect メソッドは文字通り「接続する」という意味でコードの最初に記述してください。この１行だけのプログラムを実行して CommandButton1 をクリックすると TOS9000 は REMOTE 状態になるはずです。 反対に「切り放す」は Disconnect です。

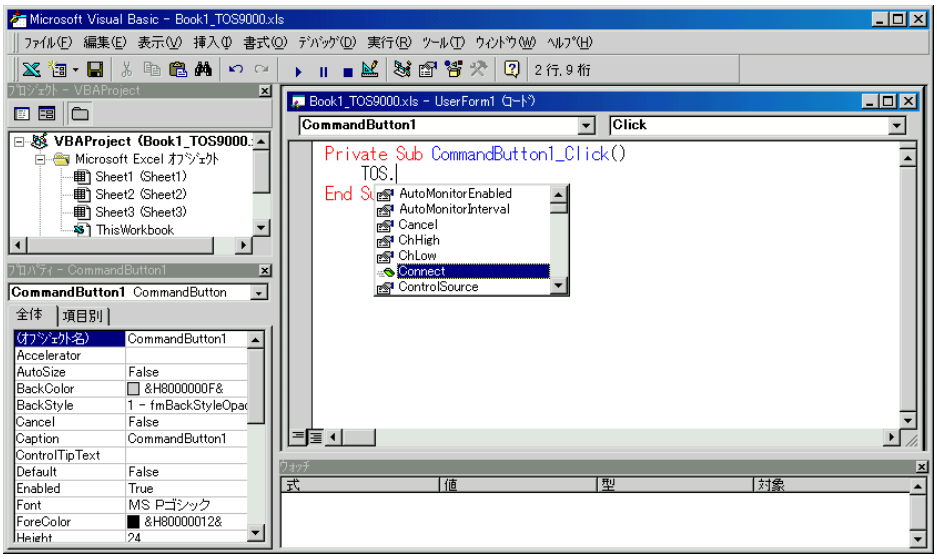

Figure 4-3 自動メンバ表示機能

従って、下記のコードのように UserForm が生成されるときに Connect して、アンロードされるときに Disconnect するのも一つの方法です。 また、CommandButton を追加し、Caption をかえて Connect ボタン, Disconnect ボタンを付けるのもよいと思います。

```
Private Sub UserForm_Initialize()
    tos.Connect "GPIB::1"
End Sub
```

```
Private Sub UserForm_Terminate()
    tos.Disconnect
End Sub
```

Connect までできれば、要領が把握できたと思います。 Help を参照しながらアプリケーションを完成させてください。

4.4 Help の参照

機器ドライバの Help ファイルの参照の仕方は、VBA の表示メニューから オブジェクト ブラウザ を表示し、プロジェクト/ライブラリ の ComboBox で "TOS9000Lib" を選択します。 Tos9000 のメンバのどれかにカーソルを合わせて ヘルプ ボタンをクリックすると TOS9000 Driver の Help が表示されます。

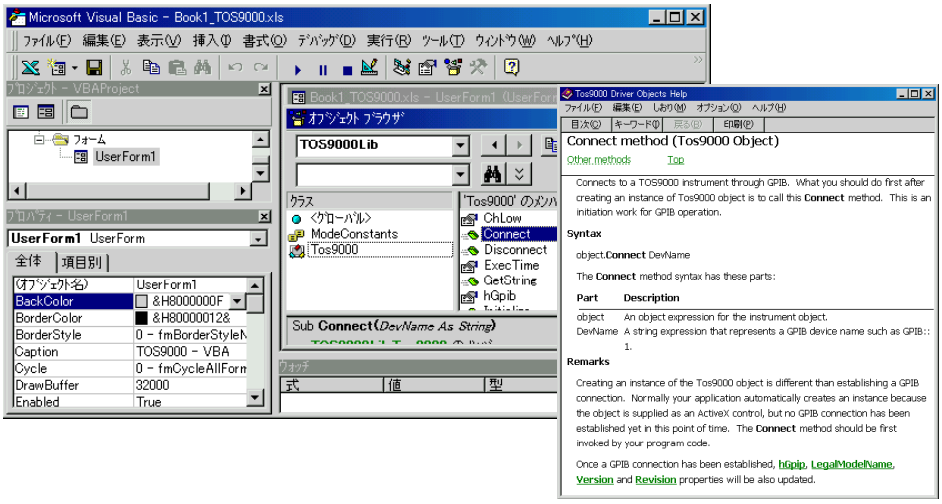

Figure 4-4　オブジェクト ブラウザによる Help の参照

4.5 プロパティページ

また、TOS9000 の機器ドライバには設計時プロパティが備わっていますので、これを活用すれば Start ボタンと Stop ボタンがあれば制御できます。 Auto Monitor タブはイベント処理のためのプロパティの設定です。

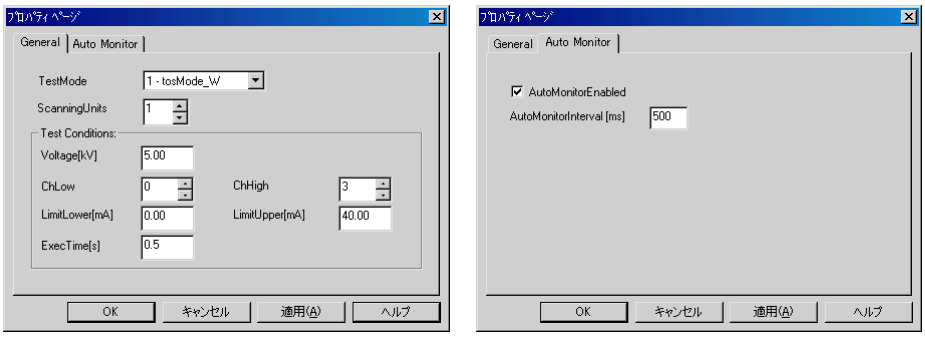

Figure 4-5 TOS9000 ドライバのプロパティページ

4.6 イベント処理

UserForm 上の TOS9000 アイコンをダブルクリックするか、コード表示の最上部のオブジェクト ComboBox を tos にして、プロシージャ ComboBox を TestingW にします。(Figure 4-6)
すると、W テスト中のイベント処理を記述する Sub プロシージャが生成されます。

```
Private Sub tos_TestingW(ByVal Status As Integer, ByVal Voltage As Double, ByVal Current As Double)


End Sub
```

Figure 4-6　VBA のｵﾌﾞｼﾞｪｸﾄとﾌﾟﾛｼｰｼﾞｬの ComboBox

このイベントを有効にするには AutoMonitorEnabled プロパティを True にする必要があります。 AutoMonitorInterval[ms]は適当な値に設定してください。 500ms 程度が良いでしょう。
サンプルコードは 4.7 項を参照してください。

4.7 サンプルコード

Excel らしくシートに試験条件を設定して、それを参照するサンプルコードを作成します。 サンプルコードは Excel シートの A,B,C の各列に設定した試験条件の内のどれかを選択できるオプションボタンを配置し、Start をクリックすると、その試験条件で試験を開始します。

耐電圧試験の試験中はリーク電流値を絶縁抵抗試験の試験中は測定抵抗値を 500ms おきに Label に表示します。

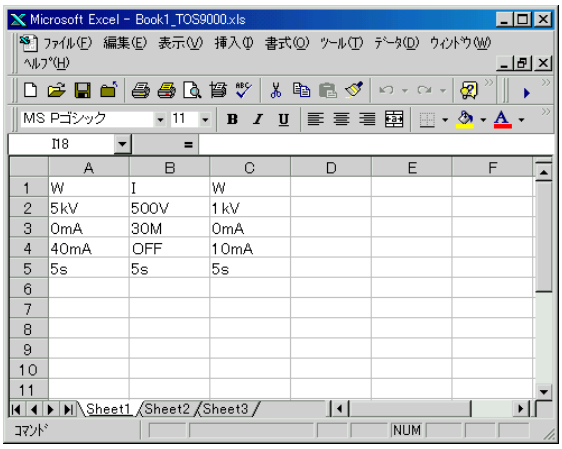

Figure 4-7　Excel のシート

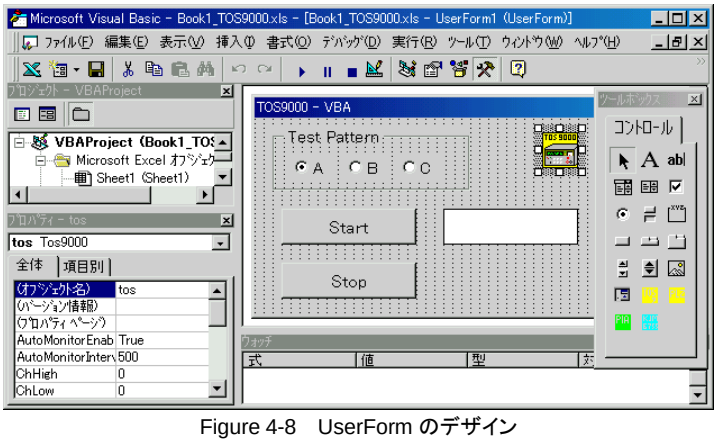

Figure 4-8　UserForm のデザイン

```
Private Sub CommandButtonStart_Click()     ' Startﾎﾞﾀﾝがｸﾘｯｸされた
  Dim szTestPat As String
  Dim szCmd As String
  Dim nSts As Integer

  If OptionButton1.Value = True Then       ' Aのﾃｽﾄを選択
    szTestPat = "A"
  ElseIf OptionButton2.Value = True Then   ' Bのﾃｽﾄを選択
    szTestPat = "B"
  Else                                     ' Cのﾃｽﾄを選択
    szTestPat = "C"
  End If

  If Worksheets("Sheet1").Range(szTestPat + "1") = "W" Then
                                           ' Excelｼｰﾄ1の1行目の文字がW
    tos.TestMode   = tosMode_W             ' 耐圧試験ﾓｰﾄﾞ
    tos.Voltage    = Val(Worksheets("Sheet1").Range(szTestPat + "2"))
    tos.LimitLower = Val(Worksheets("Sheet1").Range(szTestPat + "3"))
    tos.LimitUpper = Val(Worksheets("Sheet1").Range(szTestPat + "4"))
    tos.ExecTime   = Val(Worksheets("Sheet1").Range(szTestPat + "5"))
                         ' 2～5行目を参照して電圧，下限，上限，試験時間 を設定
  ElseIf Worksheets("Sheet1").Range(szTestPat + "1") = "I" Then
                                           ' Excelｼｰﾄ1の1行目の文字がI
    tos.TestMode   = tosMode_I             ' 絶縁試験ﾓｰﾄﾞ
    tos.Voltage    = Val(Worksheets("Sheet1").Range(szTestPat + "2"))
                     / 1000
    tos.LimitLower = Val(Worksheets("Sheet1").Range(szTestPat + "3"))
    tos.LimitUpper = Val(Worksheets("Sheet1").Range(szTestPat + "4"))
    tos.ExecTime   = Val(Worksheets("Sheet1").Range(szTestPat + "5"))
                       ' 2～5行目を参照して電圧，下限，上限，試験時間 を設定
  Else
    Exit Sub
  End If
  tos.Start                                ' Start
End Sub

Private Sub CommandButtonStop_Click()      ' Stopﾎﾞﾀﾝがｸﾘｯｸされた
    tos.Stop
End Sub

Private Sub tos_TestingI(ByVal Status As Integer, ByVal Resistance
As Double)                                 ' 絶縁試験時のｲﾍﾞﾝﾄ処理
  Dim dMomR As Double

  dMonR = Resistance
  If dMonR > 99900 Then
    Label1.Caption = "OVER"                ' 99．9GΩ以上でOVER
  Else
    Label1.Caption = Str(dMonR) + "MΩ"     ' Label1に絶縁抵抗値を表示
  End If
End Sub

Private Sub tos_TestingW(ByVal Status As Integer, ByVal Voltage As
Double, ByVal Current As Double)           ' 耐圧試験時のｲﾍﾞﾝﾄ処理
  Label1.Caption = Format(Current, "0.00mA")
End Sub

Private Sub UserForm_Initialize()          ' Form生成時の処理
  On Error GoTo GPIB_EXP                   ' 例外発生時 GPIB_EXPへ

  tos.Connect "GPIB::1"                    ' ｺﾈｸﾄ
  Exit Sub

GPIB_EXP:
  MsgBox Err.Description                   ' 例外発生時のﾒｯｾｰｼﾞﾎﾞｯｸｽ
  tos.Disconnect
  End                                      ' ｱﾌﾟﾘｹｰｼｮﾝの終了
End Sub

Private Sub UserForm_Terminate()           ' Form消滅時の処理
  tos.Disconnect                           ' ﾃﾞｨｽｺﾈｸﾄ
End Sub
```

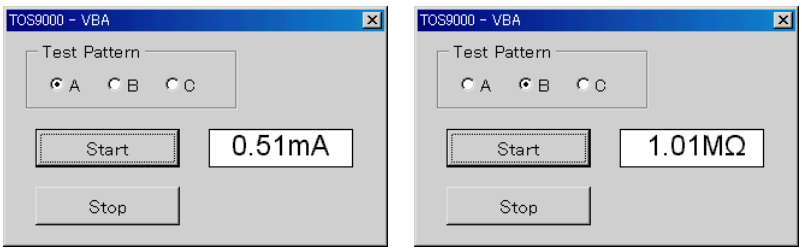

Figure 4-9　サンプルプログラムの実行

4.8 SetString メソッド, GetString メソッド

SetString メソッドはコマンドをそのまま扱えます。 ibwrt と ibrd あるいは PRINT@1; と INPUT@1; のようなものです。

```
Private Sub CommandButtonMonitor_Click()
    tos.SetString "VDATA?"
    Label1.Caption = tos.GetString
End Sub
```

第 5 章 - Visual C++ 6.0（ダイアログベースのアプリケーション）

5.1 プロジェクトの作成

ダイアログ ベースのアプリケーションのプロジェクトを作成します。［ファイル］メニューの［新規作成］から［プロジェクト］タブの［MFC AppWizard(exe)］を選択します。 プロジェクト名は［TOS9000］とします。（Figure 5-1）

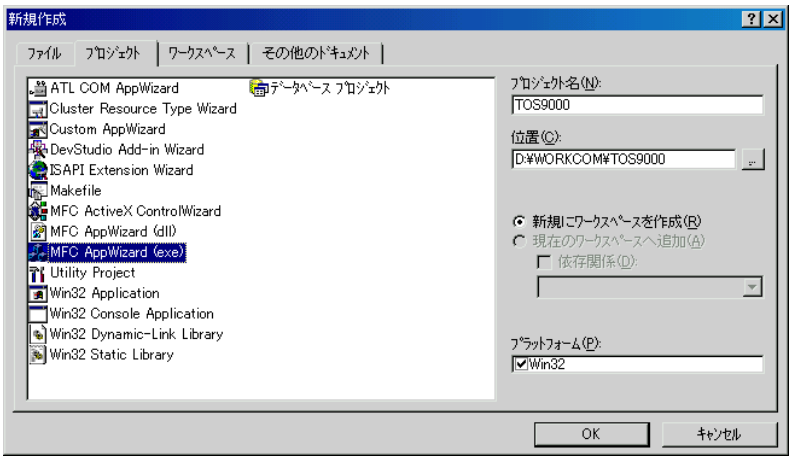

Figure 5-1　Visual C++ 6.0［新規作成］ダイアログ

MFC AppWizard - Step1 で作成するアプリケーションの種類を［ダイアログ ベース］にします。（Figure 5-2） Step2～4 はデフォルトのままです。

Figure 5-2　MFC AppWizard – Step1/4

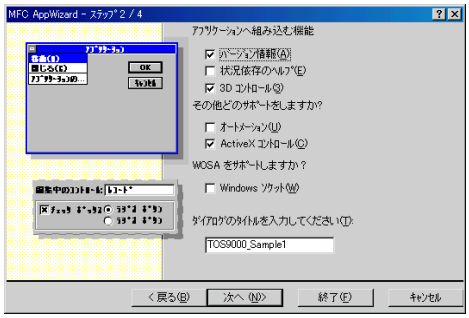

Figure 5-3　MFC AppWizard – Step2/4

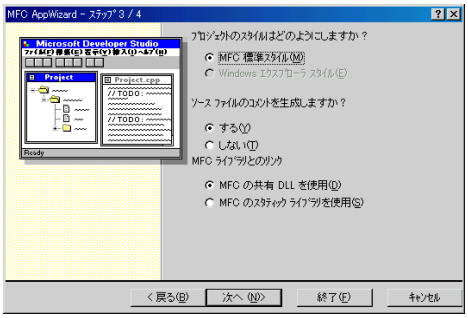

Figure 5-4　MFC AppWizard – Step3/4

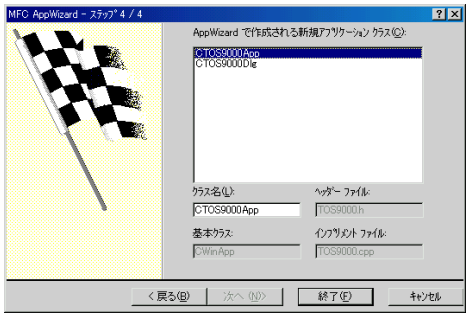

Figure 5-5　MFC AppWizard – Step4/4

5.2 コンポーネントの追加

Visual C++ の[プロジェクト]メニューから[プロジェクトへ追加]→[コンポーネントおよびコントロール]を選択し、[コンポーネントおよびコントロールギャラリ]ダイアログを表示します。（Figure 5-6）［Registered ActiveX Controls]フォルダをダブルクリックして ActiveX コントロールのリストを表示します。 機器ドライバがインストールされていれば、そのリストに **□ Kikusui xxxx Driver** が表示されているはずです。リストから目的のデバイスの機器ドライバが見つかったらカーソルを合わせて、 挿入 ボタンをクリックしてください。 ダイアログを閉じると[コントロール]ツールボックス上に機器ドライバのコンポーネントが追加されます。（Figure 5-7）

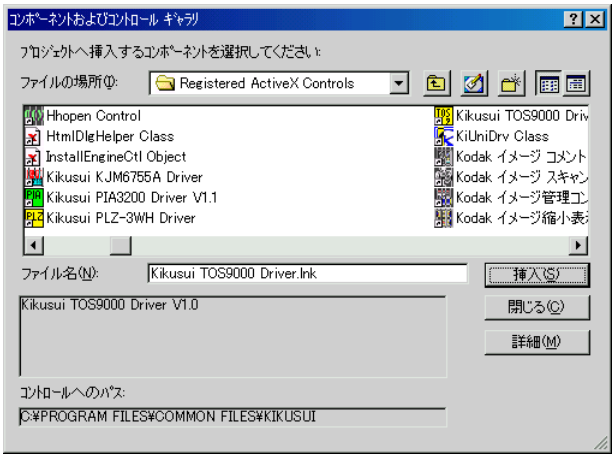

Figure 5-6 ［コンポーネントおよびコントロールギャラリ]ダイアログ

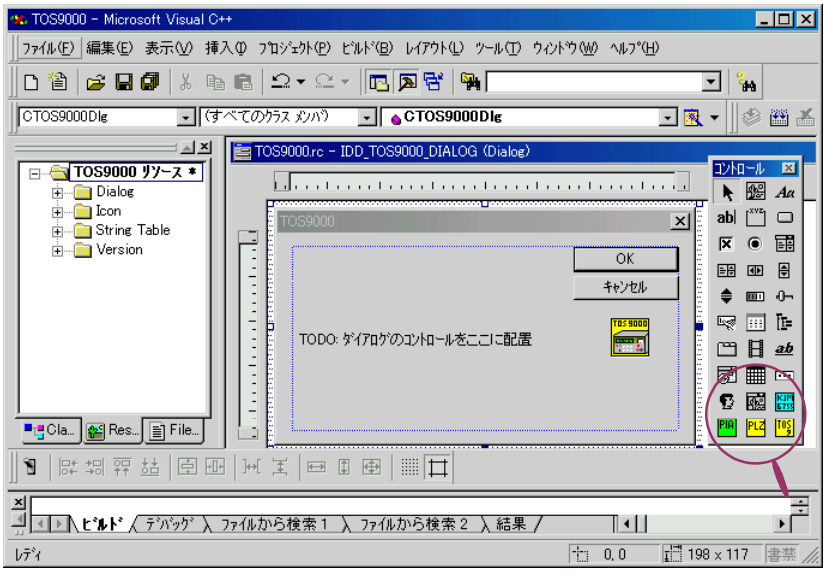

Figure 5-7　機器ドライバの ActiveX コントロール

5.3 コントロールの配置

Figure 5-7 のように[コントロール]ツールボックスから TOS9000 をダイアログウィンドウに貼り付けます。

> 注記: ここでは TOS9000 の機器ドライバを例にとりますが、他の機器ドライバをお使いになる場合でも、容易に想像可能と考えます。

デフォルトの OK, キャンセル ボタンを削除し、新たにボタンを4つ、エディット ボックスを2つ配置します。 エディット ボックスの説明のためのスタティック テキストとイベント表示のスタティック テキストを配置します。(Figure 5-8)

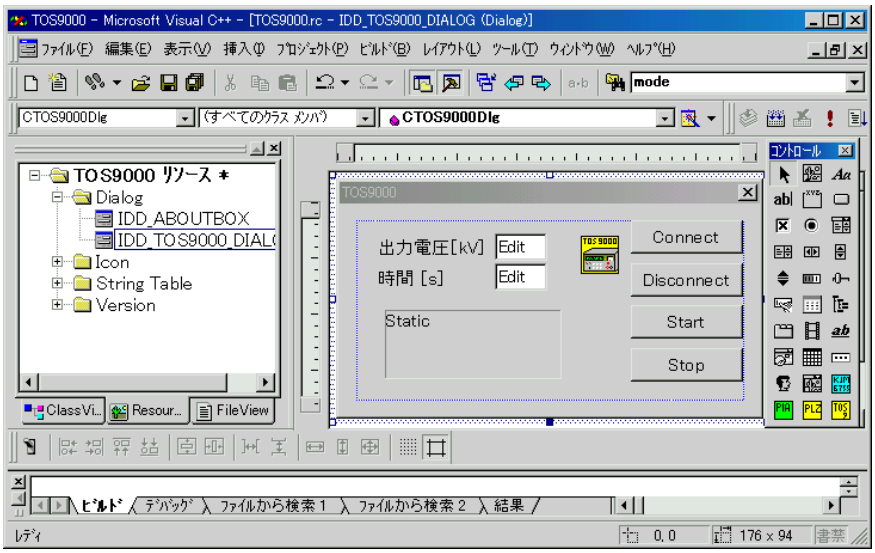

Figure 5-8 ウィンドウのデザイン

各ボタンの ID とキャプションをプロパティ ダイアログを開いて変更します。(Figure 5-9)

Figure 5-9 コントロールのプロパティ

Tos9000 Class プロパティで ID を IDC_TOS9000 とします。 MFC ClassWizard のメンバ変数タブで IDC_TOS9000 のメンバ変数 m_tos9000 を追加します。(Figure 5-10)

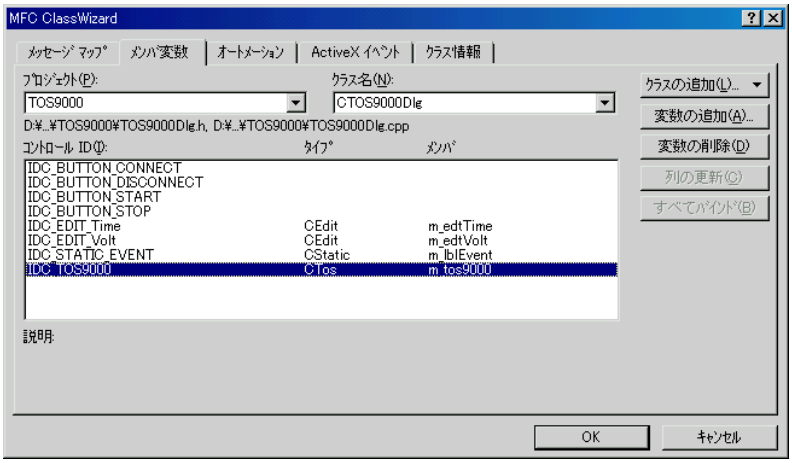

Figure 5-10 Class Wizard でメンバ変数の追加

5.4 最初のコード - Connect

Connect ボタンをダブルクリックして、Connect ボタンがクリックされたときのコードを記述します。 **m_tos9000.** まで入力すると Visual C++ 6.0 の自動補完機能によりクラスのメンバが表示されます。 co まで入力すると Connect メソッドが見付かるはずです。

try – catch の例外処理も記述しておきます。

```
void CTOS9000Dlg::OnButtonConnect()
{
    CString strVolt;

    try{
        m_tos9000.Connect("GPIB::1");
    }
    catch( COleDispatchException* pE){
        CHAR szMsg[ 64];
        pE->GetErrorMessage( szMsg, sizeof( szMsg), NULL);
        AfxMessageBox( szMsg);
        pE->Delete();
    }
}
```

Connect できなかった場合、Figure 5-11 のエラーメッセージを出します。

Figure 5-11 エラーメッセージ

5.5 イベントの処理

Form 上の TOS9000 アイコンを右クリックしてイベント ハンドラの新規作成ダイアログを開きます。 TestingW を選択して ハンドラの追加 ボタンをクリックします。(Figure 5-12)

すると、W テスト中のイベント処理を記述する関数が生成されます。

```
void CTOS9000Dlg::OnTestingWTos9000(short Status, double Voltage, double Current)
{
    // TODO: この位置にコントロール通知ハンドラ用のコードを追加してください
}
```

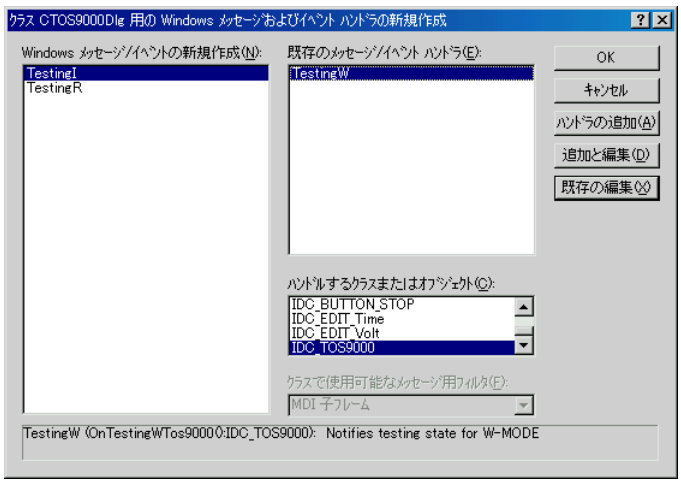

Figure 5-12 イベント ハンドラの作成

このイベントを有効にするには Tos9000 Class プロパティの Auto Monitor タブで AutoMonitor Enabled プロパティを True にする必要があります。 AutoMonitorInterval [ms]を適当な値に設定してください。 500ms 程度が良いでしょう。 5.6 項のサンプルコードでは耐電圧試験の試験中の電圧と電流を 500ms おきにスタティック テキストに表示します。

5.6 サンプルコード

Connect と同じ要領で Disconnect ボタン，Start ボタン，Stop ボタンがクリックされたときのコードを記述します。

```
void CTOS9000Dlg::OnButtonDisconnect()    // Disconnectﾎﾞﾀﾝがｸﾘｯｸされた
{
    m_tos9000.Disconnect();               // ﾃﾞｨｽｺﾈｸﾄ
}

void CTOS9000Dlg::OnButtonStart()         // Startﾎﾞﾀﾝがｸﾘｯｸされた
{
    CString strVolt;
    CString strTime;

    try{
        m_edtVolt.GetWindowText( strVolt);// Editﾎﾞｯｸｽの値を取得
        m_edtTime.GetWindowText( strTime);//        〃

        m_tos9000.SetTestMode( 1);        // 1:TosModeW, 2:TosModeI
                                          // 3:TosModeR
        m_tos9000.SetVoltage( atof( strVolt)); // 試験電圧の設定
        m_tos9000.SetLimitLower( 0);      // 下限判定値の設定
        m_tos9000.SetLimitUpper( 40);     // 上限判定値の設定
        m_tos9000.SetExecTime( atof( strTime)); // 試験時間の設定
        m_tos9000.Start();                // Start
    }
    catch( COleDispatchException* pE){    // 例外処理
        CHAR szMsg[ 64];
        pE->GetErrorMessage( szMsg, sizeof( szMsg), NULL);
                                          // ｴﾗｰﾒｯｾｰｼﾞの取得
        AfxMessageBox( szMsg);            // ﾒｯｾｰｼﾞﾎﾞｯｸｽの表示
        pE->Delete();                     // 後始末
    }
}

void CTOS9000Dlg::OnButtonStop()          // Stopﾎﾞﾀﾝがｸﾘｯｸされた
{
    m_tos9000.Stop();                     // Stop
}

void CTOS9000Dlg::OnTestingWTos9000(short Status, double Voltage,
double Current)                           // 耐圧試験中のｲﾍﾞﾝﾄ処理
{
    CString strEvent;

    strEvent.Format( "%3.2fkV¥n%3.2fmA", Voltage, Current);
    m_lblEvent.SetWindowText( strEvent);  // Static Textに電圧と電流を表示
}
```

5.7 ビルドして実行

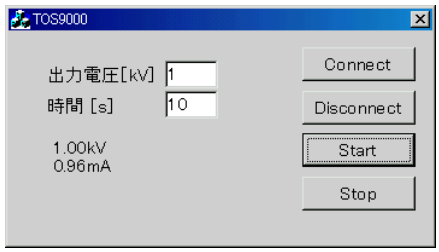

Figure 5-13 サンプルプログラムの実行

5.8 Help

メソッドやプロパティ、イベントの詳細については、各機器ドライバの Help を参照してください。

C:¥Program Files¥Common Files¥Kikusui Shared¥xxxx.hlp

第 6 章 - Visual C++ 6.0（事前バインディングによる方法）

6.1 プロジェクトの作成

ドキュメント/ビュー アーキテクチャー アプリケーション の事前バインディングによる方法を説明します。 SDI アプリケーションを例にとりますが、MDI アプリケーションでもかまいません。ダイアログウィンドウにコントロールを貼るのではなく、メニューにデバイス制御の項目を追加していくこととします。

まず、MFC AppWizard(exe)の新規プロジェクト（Tos9000SDI）を作成します。（Figure 5-1）
アプリケーションの種類は SDI とします。（Figure 6-1）

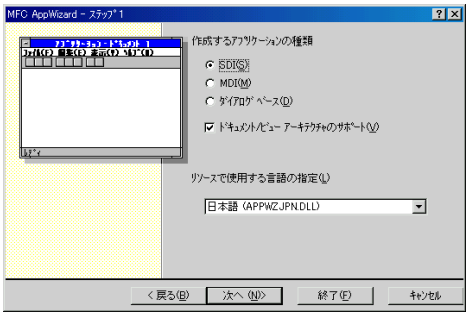

Figure 6-1 MFC AppWizard – Step1/6

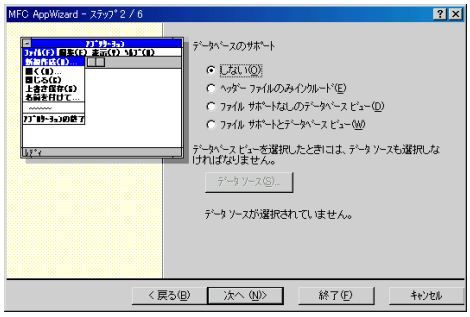

Figure 6-2 MFC AppWizard – Step2/6

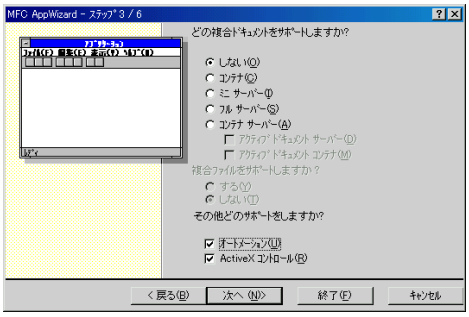

Figure 6-3 MFC AppWizard – Step3/6

Figure 6-4 MFC AppWizard – Step4/6

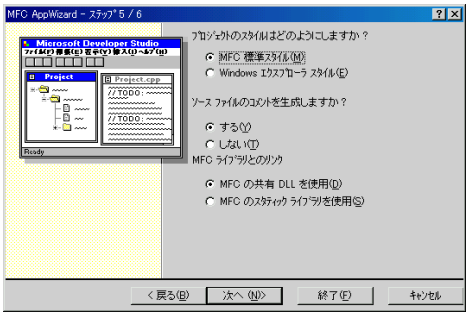

Figure 6-5 MFC AppWizard – Step5/6

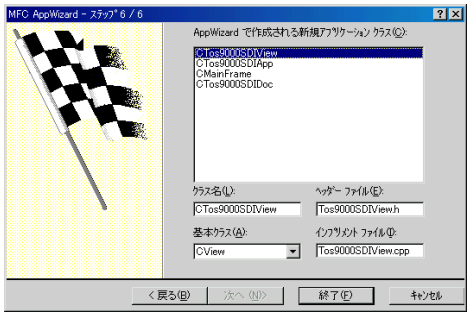

Figure 6-6 MFC AppWizard – Step6/6

6.2 タイプ ライブラリのインポート

StdAfx.h を開き、他の include 文よりも後に、次の #import 文を追加します。

「#import "C:¥Program Files¥Common Files¥Kikusui Shared¥TOS9000.OCX" named_guids no_namespace」

```
#include <afxcmn.h>        // MFC の Windows コモン コントロール サポート

#import "C:¥Program Files¥Common Files¥Kikusui Shared¥TOS9000.OCX"
        named_guids no_namespace

#endif // _AFX_NO_AFXCMN_SUPPORT
```

注記: 他の機器ドライバとクラス名が衝突してしまう可能性がある場合は # import 擬似命令の最後の no_namespace を記述しません。 パワーサプライコントローラ PIA3200 と PIA4800 シリーズを扱うアプリでは ISupply 型が名前衝突します。 no_namespace を記述しないことで、すべての CLSID, 型名(ITos9000 型など), インターフェース ID(IID_ITos9000 など)で明示的な名前空間の指定がアプリ側に求められます。

CTos9000SDIApp クラスの InitInstance を開き、AfxOleInit()の呼び出しが生成されていなければ記述してください。 AppWizard のオプションで ☑ オートメーション がチェックされていれば自動的に生成されます。(Figure 6-3)

6.3 メニューの追加

ワークスペースウィンドウの ResourceView の Menu で IDR_MAINFRAME の表示の右側に インスツルメンツ メニューを追加し、さらに、そのメニューアイテムとして コネクト を追加します。ID は ID_INSTR_CONNECT とします。 ClassWizard でこのメニューハンドラを Doc クラスの COMMAND メッセージとして 関数の追加 をクリックします。(Figure 6-7)

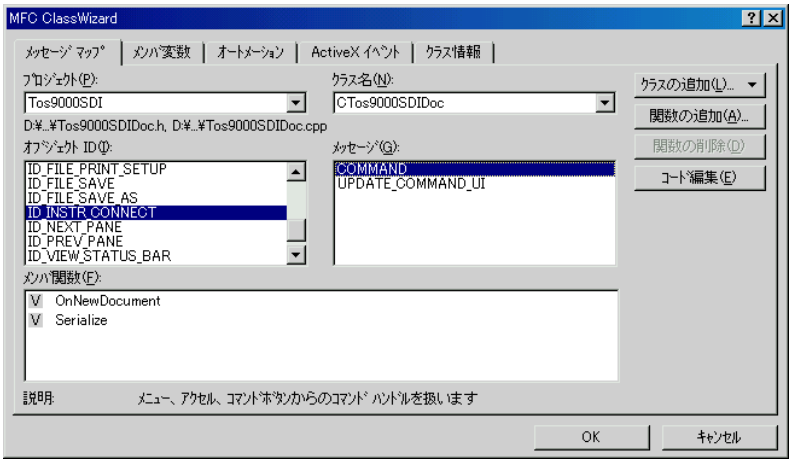

Figure 6-7 ClassWizard での関数の追加

6.4 データメンバの追加

まず、Doc クラスの宣言部にデータメンバとして "ITos9000Ptr m_spTos;" を記述します。(下記 太字部分)

```
class CTos9000SDIDoc : public CDocument
{
protected: // シリアライズ機能のみから作成します。
    ITos9000Ptr m_spTos;

    CTos9000SDIDoc();
    DECLARE_DYNCREATE(CTos9000SDIDoc)
        :
```

注記: 一度ビルドするとコンパイラが ¥Debug フォルダに自動生成する type library のファイル(拡張子 .tlh)を覗いてみると ITos9000 が Smart pointer として typedef されているのがわかるでしょう。 implementation ファイル(拡張子 .tli)も覗いてみてください。

6.5 コンストラクタへの記述

Doc クラスのコンストラクタに Tos9000 オブジェクトの生成コードを記述します。(下記 太字部分) これで Doc オブジェクトの生成と同時に Tos9000 オブジェクトが生成されます。

```
CTos9000EBDoc::CTos9000EBDoc()
{
    // TODO: この位置に1度だけ呼ばれる構築用のコードを追加してください。
    HRESULT hr =::CoCreateInstance( CLSID_Tos9000, NULL,
            CLSCTX_INPROC_SERVER, IID_ITos9000, (void**)&m_spTos);
    ASSERT( SUCCEEDED(hr));

    EnableAutomation();
    AfxOleLockApp();
}
```

6.6 コネクトメニューのコード

ClassWizard で追加した関数(OnInstrConnect)のところへ行って、下記のコードを記述します。 スマートポインタの宣言が成功していれば m_spTos-> まで入力すると自動補完機能が働くはずです。 例外処理も記述しておきます。(下記 太字部分)

```
void CTos9000SDIDoc::OnInstrConnect()
{
    // TODO: この位置にコマンド ハンドラ用のコードを追加してください
    try {
        m_spTos->Connect(L"GPIB::1");
    }
    catch( _com_error e) {
        _bstr_t strDesc = e.Description();
        AfxMessageBox( strDesc);
    }
}
```

注記: Visual C++ で事前バインディングコードの場合、文字列はすべて UNICODE で扱います。_bstr_t は UNICODE と ANSI を自動変換してくれる便利なクラスです。

6.7 ビルドして実行

インスツルメンツ メニューの コネクト をクリックするとデバイスが REMOTE 状態になります。

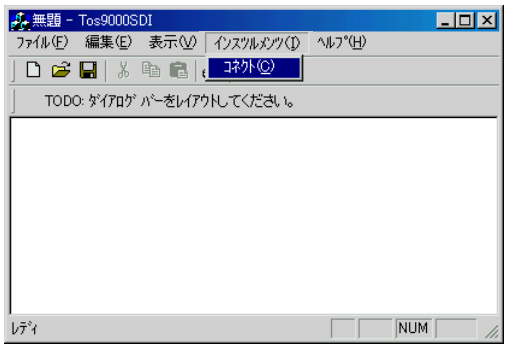

Figure 6-8 SDI アプリケーションの実行

Connect までできれば、後は想像がつくと思います。 Connect 以外のメソッドやプロパティ、イベントについては、各機器ドライバの Help を参照してください。

C:¥Program Files¥Common Files¥Kikusui Shared¥xxxx.hlp

第 7 章 - Delphi 3.0

7.1 コントロールの取り込み

[コンポーネント]メニューの[ActiveX コントロールの取り込み...]を選択すると Figure 7-1 のダイアログが開きます。 機器ドライバがインストールされていれば、”登録されているコントロール”のリストに "Kikusui xxxx Driver Vxx" が表示されているはずです。 リストから目的のデバイスの機器ドライバにカーソルを合わせて、 インストール... ボタンをクリックします。

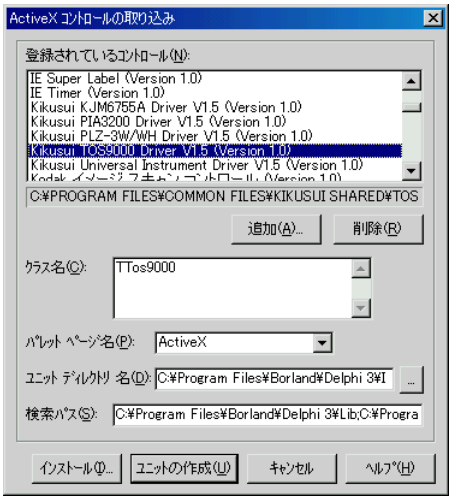

Figure 7-1 ActiveX コントロールの取り込み

[インストール]ダイアログでは[既存のパッケージへ追加]タブのまま OK ボタンをクリックします。

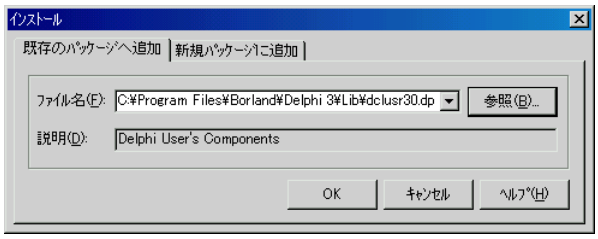

Figure 7-2 インストール 既存のパッケージへ追加

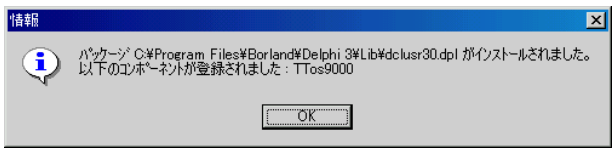

Figure 7-3 再コンパイルしてコンポーネントを登録

Delphi は自身のコンポーネントパッケージをコンパイルしコンポーネントパレットを更新します。

Figure 7-4 コンポーネントパレットの ActiveX ページ

Delphi のコンポーネントパレットの[ActiveX]ページには機器ドライバのコントロールが追加されます。

7.2 Form へのコントロールの貼付け

コンポーネントパレットの[ActiveX]ページの Tos9000 のコントロールと[Additional]ページの BitBtn を Form に貼り付けます。 BitBtn の Caption は "Connect" に、Tos9000 の Name は "tos" に代えておきます。

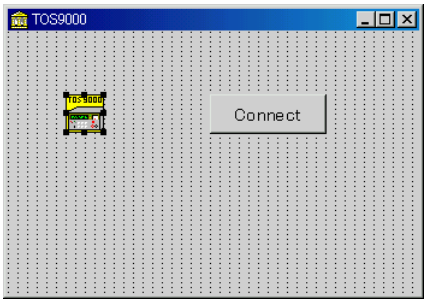

Figure 7-5　Form への貼り付け

7.3 最初のコード - Connect

Connect ボタンをダブルクリックして、Connect ボタンをクリックしたときのコードを記述します。 tos. と入力するとコード補完支援機能により Tos9000 オブジェクトのメンバが表示されます。 Connect メソッドにカーソルを合わせて Enter キーを押します。 tos. に続けて Connect と入力できます。 さらに（'GPIB::1'）とデバイス名を入力します。 'GPIB::1' は GPIB アドレスが 1 の場合です。 RS-232C では 'ASRL1' と入力します。

```
procedure TForm1.BtnConnectClick(Sender: TObject);
begin
    tos.Connect('GPIB::1');
end;
```

7.4 try - except による例外処理

try - except による例外処理を記述するには、uses に ComObj を追加する必要があります。

```
uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, Buttons, OleCtrls, TOS9000Lib_TLB, ComObj;
```

例外が発生した場合には黄色の感嘆符と OK ボタンが表示されるメッセージダイアログを表示するようにします。

```
procedure TForm1.BtnConnectClick(Sender: TObject); // Formを生成
begin
    try
        tos.Connect('GPIB::1');           // ｺﾈｸﾄ
    except
        on e: EOleException do begin      // 例外処理
            MessageDlg( e.Message, mtWarning, [mbOK], 0);
                          // 黄色の感嘆符とOKボタンが表示されるﾒｯｾｰｼﾞﾀﾞｲｱﾛｸﾞ
        end;
    end;
end;
```

それではこのプログラムを実行してみましょう。 Connect ボタンをクリックすると TOS9000 は REMOTE 状態になるはずです。 また、電源オフにして、Connect ボタンをクリックすると Delphi のデバッグ実行時エラーを表示した後、エラーメッセージダイアログが表示されます。（Figure 7-6）

Figure 7-6　エラーメッセージ

Connect までできれば、要領が把握できたと思います。 Help を参照しながらアプリケーションを完成させてください。

C:¥Program Files¥Common Files¥Kikusui Shared¥xxxx.hlp

7.5 イベント処理

オブジェクト インスペクタで tos を選択し、イベント タブの OnTestingW の右側のセルをダブルクリックします。(Figure 7-7)

Figure 7-7 イベントの生成

すると、W テスト中のイベントを記述する procedure が生成されます。

```
procedure TForm1.tosTestingW(Sender: TObject; Status: Smallint;
  Voltage, Current: Double);
begin

end;
```

このイベントを有効にするには オブジェクト インスペクタのプロパティ で AutoMonitor Enabled を True にする必要があります。 AutoMonitorInterval[ms]を適当な値に設定してください。 500ms 程度が良いでしょう。 7.6 項のサンプルコードでは耐電圧試験の試験中の電圧と電流を 500ms おきに Label に表示します。

7.6 サンプルコード

サンプルプログラムの Form のデザインは Figure 7-8 のように、試験電圧と試験時間を Edit ボックスで設定し、Start ボタンと Stop ボタンを備えます。 TestMode:1 - tosMode_W や LimitLower, LimitUpper は オブジェクト インスペクタのプロパティ であらかじめ設定しておきます。

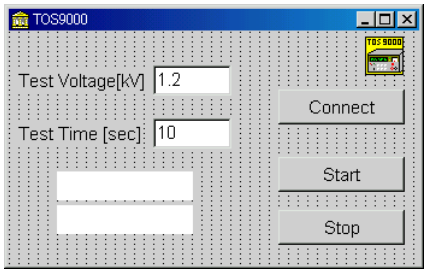

Figure 7-8　Form のデザイン

Start ボタンをクリックすると Edit ボックスの値で試験電圧と試験時間を設定し、試験を開始します。 試験中は出力電圧とリーク電流を 500ms おきに表示します。

```
procedure TForm1.BtnConnectClick(Sender: TObject); // Formを生成
begin
    try
        tos.Connect('GPIB::1');           // ｺﾈｸﾄ
    except
        on e: EOleException do begin      // 例外処理
            MessageDlg( e.Message, mtWarning, [mbOK], 0);
                                  // 黄色の感嘆符とOKボタンが表示されるメッセージダイアログ
        end;
    end;
end;

procedure TForm1.BtnStartClick(Sender: TObject);    // Startﾎﾞﾀﾝをｸﾘｯｸ
begin
    tos.Voltage  := StrToFloat(EditVoltage.Text);
                                  // Editﾎﾞｯｸｽの値を数値変換して試験電圧を設定
    tos.ExecTime := StrToFloat(EditTime.Text);
                                  // Editﾎﾞｯｸｽの値を数値変換して試験時間を設定
    tos.Start();          // Start
end;

procedure TForm1.BtnStopClick(Sender: TObject);     // Stopﾎﾞﾀﾝをｸﾘｯｸ
begin
    tos.Stop();           // Stop
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);            // Formが閉じられる
begin
    tos.Disconnect();     // ﾃﾞｨｽｺﾈｸﾄ
end;

procedure TForm1.tosTestingW(Sender: TObject; Status: Smallint;
  Voltage, Current: Double);              // 耐圧試験中のｲﾍﾞﾝﾄ処理
begin
    LabelMonitor1.Caption := Format('%2.2fkV', [Voltage]);
    LabelMonitor2.Caption := Format('%2.2fmA', [Current]);
                                          // Labelに電圧と電流を表示
end;
```

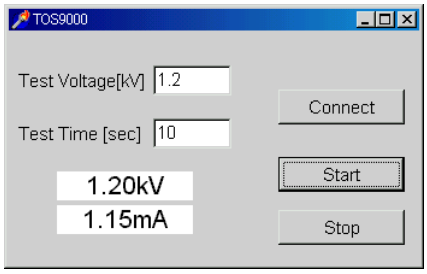

Figure 7-9　サンプルプログラムの実行